C Programming Examples

C Programming Keywords and Identifiers

Character set

Character set are the set of alphabets, letters and some special characters that are valid in C language.

Alphabets:

Uppercase: A B C ..... X Y Z

Lowercase: a b c ..... x y z

Digits:

0 1 2 3 4 5 6 8 9

Special Characters:

Special Characters in C language

,	<	>	.	_	(	)	;	$	:	%	[	]	#	?
'	&	{	}	"	^	!	*	/	|	-	\	~	+	

White space Characters:

blank space, new line, horizontal tab, carriage return and form feed

Keywords:

Keywords are the reserved words used in programming. Each keywords has fixed meaning and that cannot be changed by user. For example:

```
int money;
```

Here, int is a keyword that indicates, `'money'` is of type integer.

Praveen Shandilya

As, C programming is case sensitive, all keywords must be written in lowercase. Here is the list of all keywords predefined by ASCII C.

32 Keywords in C Programming Language

auto	**double**	**int**	**struct**
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Identifiers

In C programming, identifiers are names given to C entities, such as variables, functions, structures etc. Identifier are created to give unique name to C entities to identify it during the execution of program. For example:

```
int money;
int mango_tree;
```

Here, `money` is a identifier which denotes a variable of type integer. Similarly, `mango_tree` is another identifier, which denotes another variable of type integer.

Rules for writing identifier

1.An identifier can be composed of letters (both uppercase and lowercase letters), digits and underscore '_' only.
2.The first letter of identifier should be either a letter or an underscore. But, it is discouraged to start an identifier name with an underscore though it is legal. It is because, identifier that starts with underscore can conflict with system names. In such cases, compiler will complain about it. Some system names that start with underscore are `_fileno`, `_iob`, `_wfopen` etc.

Praveen Shandilya

3.There is no rule for the length of an identifier. However, the first 31 characters of an identifier are discriminated by the compiler. So, the first 31 letters of two identifiers in a program should be different.

Flowchart

1. Graphical representation of any program is called flowchart.
2.There are some standard graphics that are used in flowchart as following:

Figure: Start/Stop terminal box

Figure: Input/Output box

Figure: Process/Instruction box

Figure: Lines or Arrows

Figure: Decision box

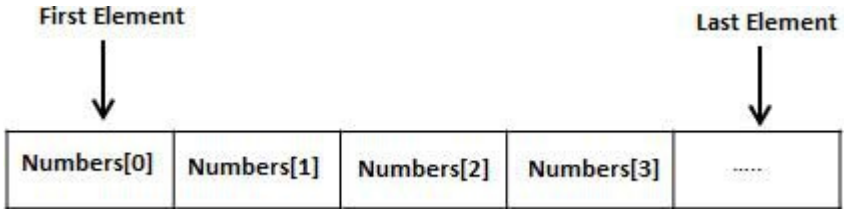
Figure: Connector box

Figure: Comment box

Figure: Preparation box

Figure: Separate box

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Declaring Arrays

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows:

```
type arrayName [ arraySize ];
```

This is called a *single-dimensional* array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid C data type. For example, to declare a 10-element array called **balance** of type double, use this statement:

```
double balance[10];
```

Now *balance* is avariable array which is sufficient to hold upto 10 double numbers.

Initializing Arrays

You can initialize array in C either one by one or using a single statement as follows:

```
double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].

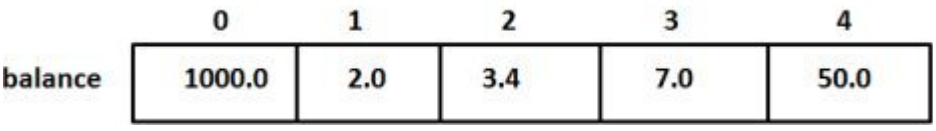
If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write:

```
double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array:

```
balance[4] = 50.0;
```

The above statement assigns element number 5th in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called base index and last index of an array will be total size of the array minus 1. Following is the pictorial representation of the same array we discussed above:



Types of C arrays:

There are 2 types of C arrays. They are,

1.One dimensional array
2.Multidimensional array
•Two dimensional array
•Three dimensional array, four dimensional array etc

1. One dimensional array in C:

•Syntax : data-type arr_name[array_size];

Array declaration	Array initialization	Accessing array
Syntax: data_type arr_name [arr_size];	data_type arr_name [arr_size]= (value1, value2, value3,....);	arr_name[index];
int age [5];	int age[5]={0, 1, 2, 3, 4, 5};	age[0]; /*0 is accessed*/ age[1]; /*1 is accessed*/ age[2]; /*2 is accessed*/
char str[10];	Char str[10]={'H','a','i'}; (or) char str[0] = 'H'; char str[1] = 'a'; char str[2] = 'i;	str[0]; /*H is accessed*/ str[1]; /*a is accessed*/ str[2]; /* i is accessed*/

Example program for one dimensional array in C.

```
1. #include<stdio.h>
2. int main()
3. {
4.         int I;
5.         int arr[5] = {10,20,30,40,50};
6.         / /declaring and Initializing array in C
7.         //To initialize all array elements to 0, use int arr[5]={0};
```

Praveen Shandilya

```
8.          /* Above array can be initialized as below also
9.          arr[0] = 10;
10.                 arr[1] = 20;
11.                 arr[2] = 30;
12.                 arr[3] = 40;
13.                 arr[4] = 50;
14.                 */
15.                 for  (i=0;i<5;i++)
16.                 {
17.                 // Accessing each variable
18.                 printf("value of arr[%d] is %d \n",  i,  arr[i]);
19.           }
20.         }
```

Output

```
value of arr[0] is 10
value of arr[1] is 20
value of arr[2] is 30
value of arr[3] is 40
value of arr[4] is 50
```

2.Two dimensional array in C:

•Two dimensional array is nothing but array of array.

•syntax : data_type array_name[num_of_rows][num_of_column]

S.no	Array declaration	Array initialization	Accessing array
1	Syntax: data_type arr_name [num_of_rows] [num_of_column];	data_type arr_name[2][2] = {{0,0}, {0,1},{1,0},{1,1}};	arr_name[index];

2	Example:
	int arr[2][2];
		int arr[2][2] = {1,2, 3, 4};	arr [0] [0] = 1;
			arr [0]]1] = 2;
			arr [1][0] = 3;
			arr [1] [1] = 4;

Example program for two dimensional array in C:

```
#include<stdio.h>
int  main()
  {
int  i,j;
// declaring and Initializing array
```

Praveen Shandilya

```
int  arr[2][2]  =  {10,20,30,40};
/* Above array can be initialized as below also
arr[0][0] = 10; // Initializing array
arr[0][1] = 20;
arr[1][0] = 30;
 arr[1][1] = 40;
*/
for  (i=0;i<2;i++)
  {
    for  (j=0;j<2;j++)
     {
// Accessing variables
printf("value of arr[%d] [%d] : %d\n",i,j,arr[i][j]);

  }
 }
}
```

Output

```
value of arr[0] [0] is 10
value of arr[0] [1] is 20
value of arr[1] [0] is 30
value of arr[1] [1] is 40
```

Calculate Sum & Average of an Array

```
21./*
22.         * C program to read N integers into an array A and
23.         * a) Find the sum of negative numbers
24.         * b) Find the sum of positive numbers
25.         * c) Find the average of all numbers
26.         * Display the results with suitable headings
27.         */
28.        #include  <stdio.h>
29.        #define MAXSIZE 10
30.
31.        void  main()
32.        {
33.             int  array[MAXSIZE];
34.             int  i, num, negative_sum  =  0, positive_sum  =  0;
35.             float  total  =  0.0, average;
36.
37.             printf  ("Enter the value of N \n");
38.             scanf("%d",  &num);
39.             printf("Enter %d numbers (-ve, +ve and zero) \n",  num);
```

```
40.         for (i = 0; i < num; i++)
41.         {
42.             scanf("%d", &array[i]);
43.         }
44.         printf("Input array elements \n");
45.         for (i = 0; i < num; i++)
46.         {
47.             printf("%+3d\n", array[i]);
48.         }
49.         /* Summation starts */
50.         for (i = 0; i < num; i++)
51.         {
52.             if (array[i] < 0)
53.             {
54.                 negative_sum = negative_sum + array[i];
55.             }
56.             else if (array[i] > 0)
57.             {
58.                 positive_sum = positive_sum + array[i];
59.             }
60.             else if (array[i] == 0)
61.             {
62.                 ;
63.             }
64.             total = total + array[i] ;
65.         }
66.         average = total / num;
67.         printf("\n Sum of all negative numbers =  %d\n", negative_sum);
68.         printf("Sum of all positive numbers =  %d\n", positive_sum);
69.         printf("\n Average of all input numbers =  %.2f\n", average);
70.     }
```

Output

```
Enter the value of N
10
Enter 10 numbers (-ve, +ve and zero)
-8
9
-100
-80
90
```

Praveen Shandilya

```
45
-23
-1
0
16
Input array elements
 -8
 +9
-100
-80
+90
+45
-23
 -1
 +0
+16
Sum of all negative numbers =  -212
Sum of all positive numbers =  160
Average of all input numbers =  -5.20
```

Find the Largest Two Numbers in a given Array

```
1. /*
2.  * C program to read in four integer numbers into an array and find the
3.  * average of largest two of the given numbers without sorting the array.
4.  * The program should output the given four numbers and the average.
5.  */
6. #include <stdio.h>
7. #define MAX 4
8.
9. void main()
10.     {
11.         int array[MAX], i, largest1, largest2, temp;
12.
13.         printf("Enter %d integer numbers \n", MAX);
14.         for (i = 0; i < MAX; i++)
15.         {
16.             scanf("%d", &array[i]);
17.         }
```

```
18.
19.         printf("Input interger are \n");
20.         for (i = 0; i < MAX; i++)
21.         {
22.           printf("%5d", array[i]);
23.         }
24.         printf("\n");
25.         /* assume first element of array is the first larges t*/
26.         largest1 = array[0];
27.         /* assume first element of array is the second largest */
28.         largest2 = array[1];
29.         if (largest1 < largest2)
30.         {
31.             temp = largest1;
32.             largest1 = largest2;
33.             largest2 = temp;
34.         }
35.         for (i = 2; i < 4;  i++)
36.         {
37.             if (array[i] >= largest1)
38.             {
39.                 largest2 = largest1;
40.                 largest1 = array[i];
41.             }
42.             else if (array[i] > largest2)
43.             {
44.                 largest2 = array[i];
45.             }
46.         }
47.         printf("n%d is the first largest \n", largest1);
48.         printf("%d is the second largest \n", largest2);
49.         printf("nAverage of %d and %d = %d \n", largest1, largest2,
50.     (largest1 + largest2) / 2);
51.     }
```

Output

```
Enter 4 integer numbers
80
23
```

Praveen Shandilya

```
79
58
Input interger are
80   23   79   58

80 is the first largest
79 is the second largest

Average of 80 and 79 = 79
```

Calculate the Sum of the Array Elements using Pointer

```
1. /*
2.  * C program to read N integers and store them in an array A.
3.  * Find the sum of all these elements using pointer.
4.  */
5. #include <stdio.h>
6. #include <malloc.h>
7.
8. void main()
9. {
10.         int i, n, sum = 0;
11.         int *a;
12.
13.         printf("Enter the size of array A \n");
14.         scanf("%d", &n);
15.         a = (int *) malloc(n * sizeof(int));
16.         printf("Enter Elements of First List \n");
17.         for (i = 0; i < n; i++)
18.         {
19.             scanf("%d", a + i);
```

```
        }
        </* Compute the sum of all elements in the given array */
        for (i = 0; i < n; i++)
        {
            sum = sum + *(a + i);
        }
        printf("Sum of all elements in array = %d\n", sum);
    }
```

Output

```
Enter the size of array A
5
Enter Elements of First List
4
9
10
56
100
Sum of all elements in array = 179
```

Put Even & Odd Elements of an Array in 2 Separate Arrays

```
/*
 * C Program to accept N integer number and store them in an array AR.
 * The odd elements in the AR are copied into OAR and other elements
 * are copied into EAR. Display the contents of OAR and EAR.
 */
#include <stdio.h>

void main()
{
        long int ARR[10], OAR[10], EAR[10];
        int i, j = 0, k = 0, n;

        printf("Enter the size of array AR \n");
        scanf("%d", &n);
        printf("Enter the elements of the array \n");
        for (i = 0; i < n; i++)
```

```
17.         {
18.             scanf("%ld", &ARR[i]);
19.             fflush(stdin);
20.         }
21.         /* Copy odd and even elements into their respective arrays */
22.         for (i = 0; i < n; i++)
23.         {
24.             if (ARR[i] % 2 == 0)
25.             {
26.                 EAR[j] = ARR[i];
27.                 j++;
28.             }
29.             else
30.             {
31.                 OAR[k] = ARR[i];
32.                 k++;
33.             }
34.         }
35.         printf("The elements of OAR are \n");
36.         for (i = 0; i < j; i++)
37.         {
38.             printf("%ld\n", OAR[i]);
39.         }
40.         printf("The elements of EAR are \n");
41.         for (i = 0; i < k; i++)
42.         {
43.             printf("%ld\n", EAR[i]);
44.         }
45.     }
```

Output

```
Enter the size of array AR
6
Enter the elements of the array
34
56
78
90
12
```

Praveen Shandilya

```
39
The elements of OAR are
39
1
32768
11542516
11210377
The elements of EAR are
34
```

Sort the Array in an Ascending Order

```
1. /*
2.  * C program to accept N numbers and arrange them in an ascending order
3.  */
4. #include <stdio.h>
5.
6. void main()
7. {
8.     int i, j, a, n, number[30];
9.
10.         printf("Enter the value of N \n");
11.         scanf("%d", &n);
12.         printf("Enter the numbers \n");
13.         for (i = 0; i < n; ++i)
14.             scanf("%d", &number[i]);
15.         for (i = 0; i < n; ++i)
16.         {
17.             for (j = i + 1; j < n; ++j)
18.             {
19.                 if (number[i] > number[j])
20.                 {
21.                     a =  number[i];
22.                     number[i] = number[j];
23.                     number[j] = a;
24.                 }
25.             }
```

```
26.          }
27.          printf("The numbers arranged in ascending order are given below \n");
28.          for (i = 0; i < n; ++i)
29.               printf("%d\n", number[i]);
30.     }
```

Output

```
Enter the value of N
6
Enter the numbers
3
78
90
456
780
200
The numbers arranged in ascending order are given below
3
78
90
200
456
780
```

Sort the Array in Descending Order

```
1. /*
2.  * C program to accept a set of numbers and arrange them
3.  * in a descending order
4.  */
5. #include <stdio.h>
6.
7. void main ()
8. {
9.     int number[30];
10.          int i, j, a, n;
11.
```

```
12.         printf("Enter the value of N\n");
13.         scanf("%d", &n);
14.         printf("Enter the numbers \n");
15.         for (i = 0; i < n; ++i)
16.         scanf("%d", &number[i]);
17.         /* sorting begins ... */
18.         for (i = 0; i < n; ++i)
19.         {
20.             for (j = i + 1; j < n; ++j)
21.             {
22.                 if (number[i] < number[j])
23.                 {
24.                     a = number[i];
25.                     number[i] = number[j];
26.                     number[j] = a;
27.                 }
28.             }
29.         }
30.         printf("The numbers arranged in descending order are given below\n");
31.         for (i = 0; i < n; ++i)
32.         {
33.             printf("%d\n", number[i]);
34.         }
35.     }
```

Output

```
Enter the value of N
5
Enter the numbers
234
780
130
56
90
The numbers arranged in descending order are given below
780
234
130
90
56
```

Praveen Shandilya

Sort the N Names in an Alphabetical Order

```
1. /*
2.  * C program to read N names, store them in the form of an array
3.  * and sort them in alphabetical order. Output the given names and
4.  * the sorted names in two columns side by side.
5.  */
6. #include <stdio.h>
7. #include <string.h>
8.
9. void main()
10.     {
11.         char name[10][8], tname[10][8], temp[8];
12.         int i, j, n;
13.
14.         printf("Enter the value of n \n");
15.         scanf("%d", &n);
16.         printf("Enter %d names n", \n);
17.         for (i = 0; i < n; i++)
18.         {
19.             scanf("%s", name[i]);
20.             strcpy(tname[i], name[i]);
21.         }
22.         for (i = 0; i < n - 1 ; i++)
23.         {
24.             for (j = i + 1; j < n; j++)
25.             {
26.                 if (strcmp(name[i], name[j]) > 0)
27.                 {
28.                     strcpy(temp, name[i]);
29.                     strcpy(name[i], name[j]);
30.                     strcpy(name[j], temp);
31.                 }
32.             }
33.         }
34.         printf("\n---------------------------------------\n");
35.         printf("Input NamestSorted names\n");
36.         printf("---------------------------------------\n");
37.         for (i = 0; i < n; i++)
38.         {
```

```
39.             printf("%s\t\t%s\n", tname[i], name[i]);
40.         }
41.         printf("---------------------------------------\n");
42.     }
```

Output

```
Enter the value of n
7
Enter 7 names
g
d
q
o
c
p
e

-------------------------------------
Input Names     Sorted names
-------------------------------------
g                 c
d                 d
q                 e
o                 g
c                 o
p                 p
e                 q
```

Accept an Array & Swap Elements using Pointers

```
1.  * C program to accept an array of 10 elements and swap 3rd element
2.  * with 4th element using pointers and display the results.
3.  */
4. #include <stdio.h>
5. void swap34(float *ptr1, float *ptr2);
6.
```

```
7. void main()
8. {
9.     float x[10];
10.         int i, n;
11.
12.         printf("How many Elements...\n");
13.         scanf("%d", &n);
14.         printf("Enter Elements one by one\n");
15.         for (i = 0; i < n; i++)
16.         {
17.             scanf("%f", x + i);
18.         }
19.         /* Function call:Interchanging 3rd element by 4th */
20.         swap34(x + 2, x + 3);
21.         printf("\nResultant Array...\n");
22.         for (i = 0; i < n; i++)
23.         {
24.             printf("X[%d] = %f\n", i, x[i]);
25.         }
26.     }
27./* Function to swap the 3rd element with the 4th element in the array */
28.     void swap34(float *ptr1, float *ptr2 )
29.     {
30.         float temp;
31.         temp = *ptr1;
32.         *ptr1 = *ptr2;
33.         *ptr2 = temp;
34.     }
```

Output

```
How many Elements...
4
Enter Elements one by one
23
67
45
15

Resultant Array...
```

Praveen Shandilya

x[0] = 23.000000
x[1] = 67.000000
x[2] = 15.000000
x[3] = 45.000000